Towards Automated Detection of Struggling Student Programmers

Sanjita Patwardhan & Ruben Acuña

Educational Data Mining | Exploratory Data Analysis | Struggle

In programming courses, it is often difficult for instructors to identify students who struggle while coding. Fortunately, automated assessment tools used in courses provide a way to capture data about programming activity. Using this data source provides the foundation for developing a machine learning model to automatically classify students who are struggling, even in large courses. Such a model would help instructors target interventions to help struggling students. In this paper, we provide a step towards creating this model. We have conducted preliminary work focused on identifying features with the potential to indicate struggle, performing feature engineering to extract them, and then conducting an exploratory data analysis on real data to visualize outliers and assess feasibility.

Introduction

Identifying struggling students in computing courses remains a persistent challenge for many instructors. Students working on assignments at home are unobserved by an instructor, and even those completing an in-class activity may not catch the instructor's attention in a large class. However, courses often make use of automated assessment tools (AATs), which allow students to submit assignments and receive feedback on which parts of their assignments have correct functionality by executing a set of instructor-developed tests (Messer et al., 2024). AATs let us implicitly observe students by capturing development activity, such as submission attempts, compilation status, and progress according to test cases. There is extensive work on educational data mining (see Inhantola et al., 2015), with prior research on block-based languages (BBLs) showing how to detect struggling learners and time interventions (e.g., Dong et al., 2021; Tabarsi et al., 2022). However, much remains unknown about which patterns are most indicative of struggle and how they are shown in real datasets. When instructors can identify struggling students earlier, they are able to direct support, enhance activities, and help prevent disengagement. Without automated detection, many patterns of process struggle remain hidden unless reported by students. Unlike work with BBLs, we examine data for a general-purpose language (Java), requiring feature engineering before conducting exploratory data analysis (EDA) on submission logs. Our guiding research question is: can the features in student code traces predict whether a student will get stuck on an assignment? In this preliminary work, we investigate features such as the number of submissions each student makes, the coding time gaps, and temporal patterns of work. This work was structured as a nested Learning Engineering cycle that focused on identifying relevant data features within the challenge stage of our project to identify struggling students.

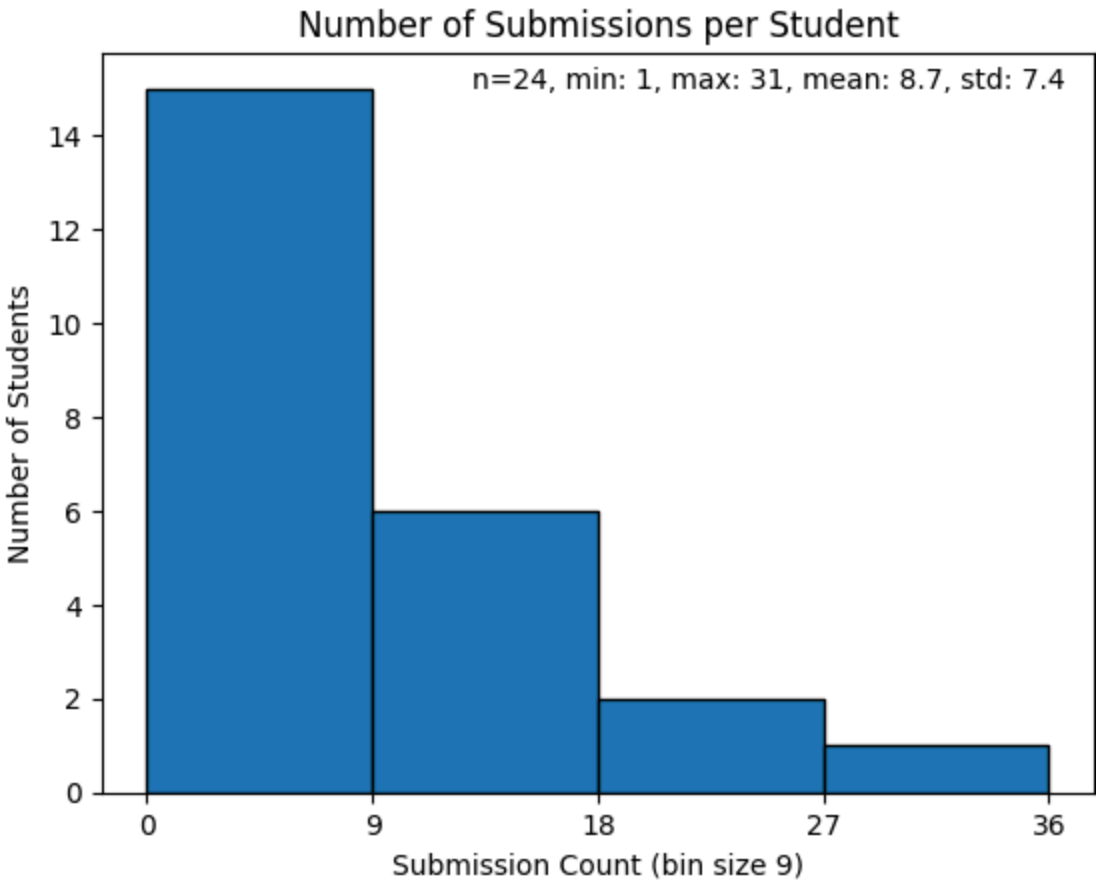
Methods

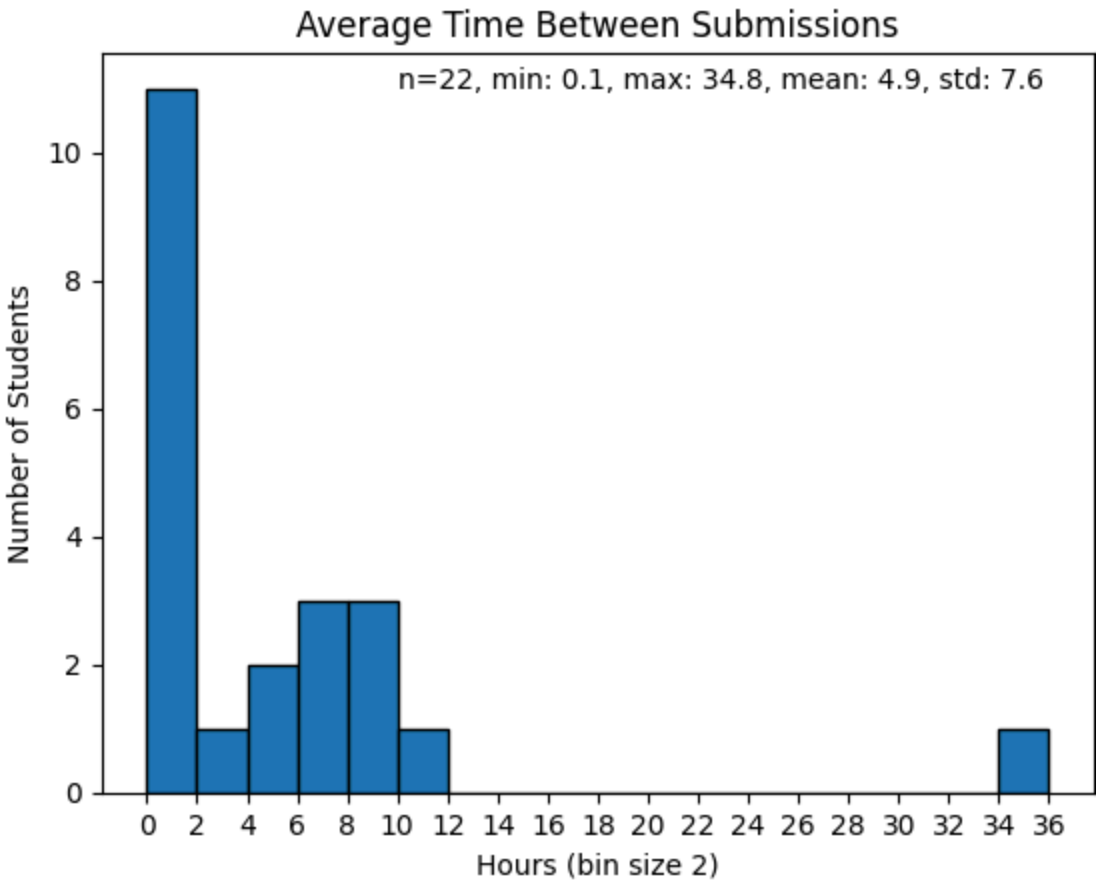
To investigate which features may signal student struggle, we implemented a data analysis pipeline that extracts features from Gradescope logs. Our script processes students' timestamped submission history and computes measures that reflect coding behaviors. We focused on four features that represent student submission behavior: 1) Number of Submissions: The number of times each student submitted during the assignment. 2) Time Between Submissions: For each student, we compute the time between consecutive submissions. The longer gaps may indicate debugging times or a break. 3) Time of Day of Work: Using the attempt timestamps, we determine the hour(s) in which work occurred. This helps reveal working patterns, such as last-minute bursts. 4) Total Coding Time: We calculate the delta between a student's first and last submission. This approximates a lower bound for the time each student engaged with the assignment as seen by AAT activity. These features were selected because we have informally observed students repeatedly submitting code to ensure it matches the assignment requirements.

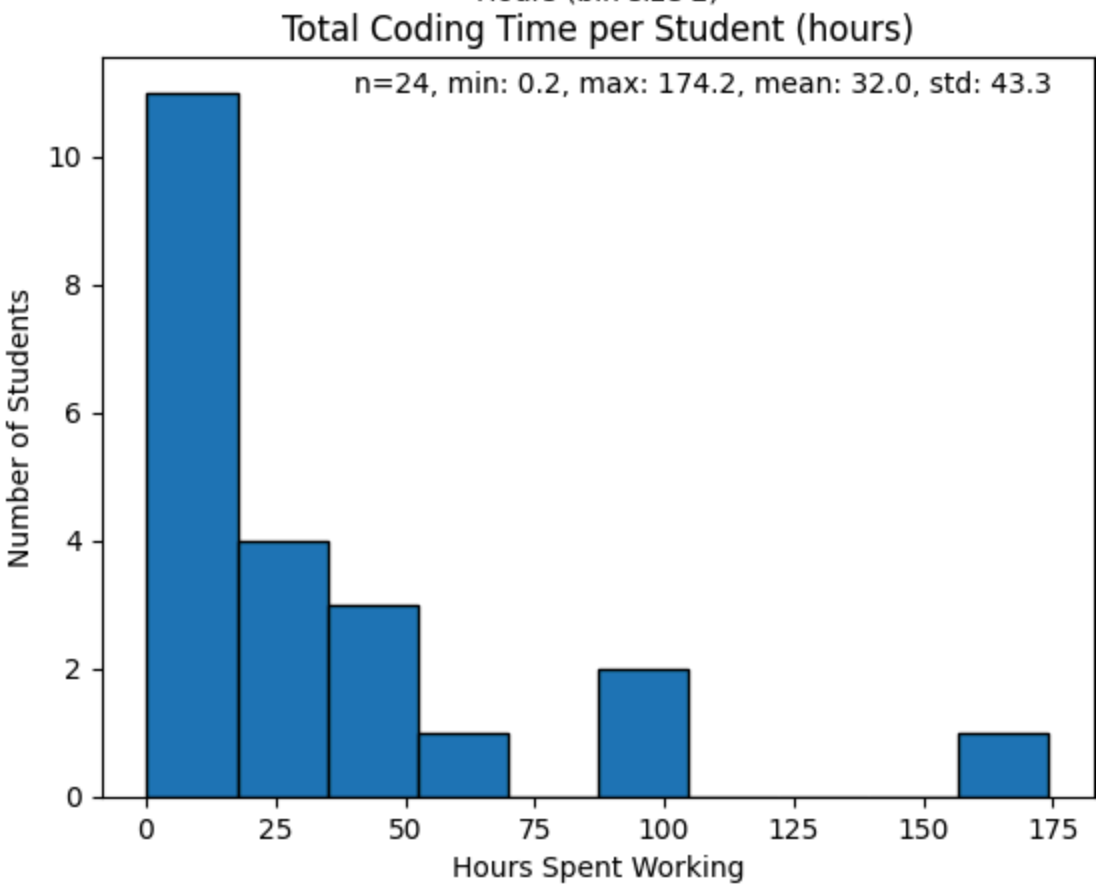
Results

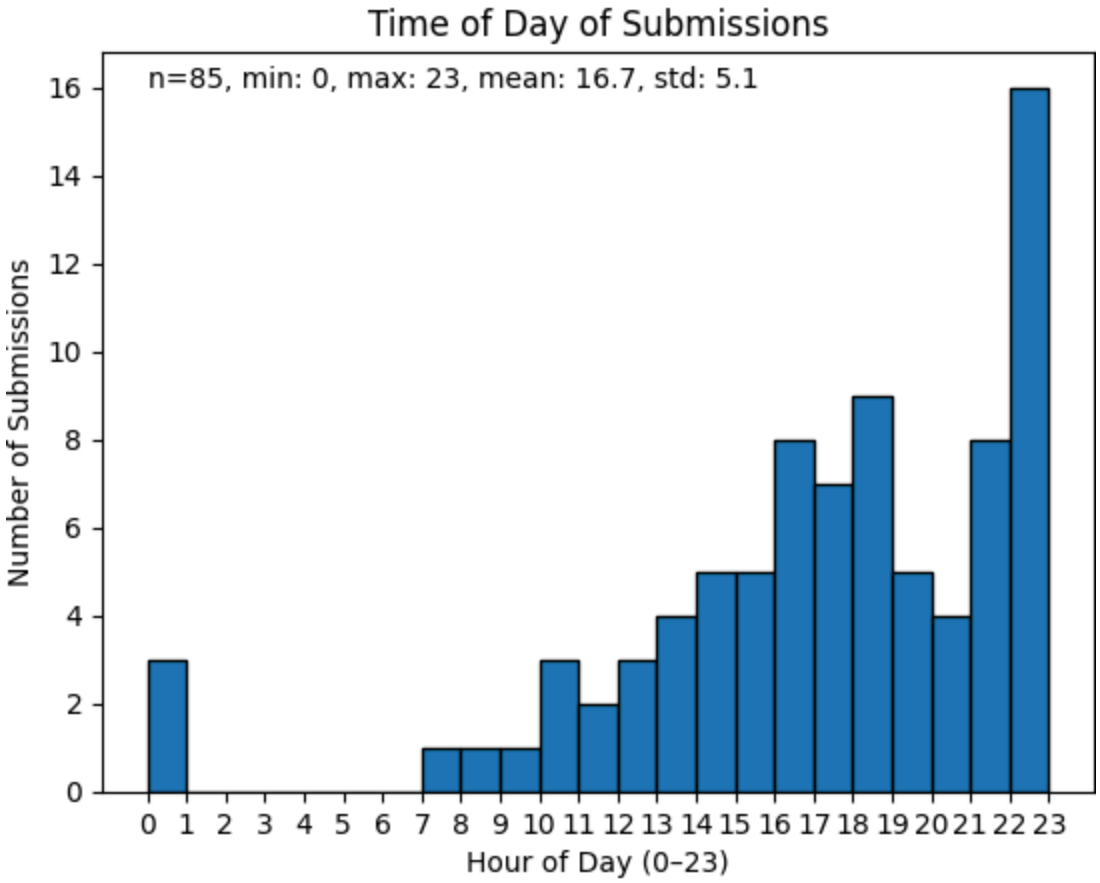
We applied our methodology to data from a face-to-face sophomore-level course on data structures & algorithms (described in Acuña & Bansal, 2024) taught in the spring of 2025. The course is taught in Java and uses an AAT to assess assignments on topics such as lists, trees, hash tables, and graphs. This paper focuses on a warm-up assignment in which students implemented a matrix class (n=24). Histograms for the features are shown in Figure 1.

Figure. 1. Visualizations for four extracted features on data structures & algorithm dataset.


Number of Submissions per Student
n=24, min: 1, max: 31, mean: 8.7, std: 7.4
Number of Students
0
2
4
6
8
10
12
14
0
9
18
27
36
Submission Count (bin size 9)


Average Time Between Submissions
n=22, min: 0.1, max: 34.8, mean: 4.9, std: 7.6
Number of Students
0
2
4
6
8
10
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
Hours (bin size 2)


Total Coding Time per Student (hours)
n=24, min: 0.2, max: 174.2, mean: 32.0, std: 43.3
Number of Students
0
2
4
6
8
10
0 25 50 75 100 125 150 175
Hours Spent Working



These plots indicate several trends. Submission counts ranged from single attempts to 31, reflecting different strategies from students and how they went about completing the assignment. Average time between submissions displayed more students working on the coding assignment with few breaks in between, with most students submitting frequently and a small number exhibiting long pauses that may indicate extended debugging. Total coding time also varied widely, with some students completing work quickly while others engaged across longer multi-hour spans. Time-of-day patterns showed clustering around late afternoon and evening hours, with many submitting a couple of hours before the 11:59 pm deadline. Based on the extracted features (and others), we plan to develop a machine learning classifier.

Discussion

Our findings suggest features that may meaningfully reflect students' coding processes. The distributions we observed, especially in submission counts, timing gaps, and total coding time, highlight that students do not engage with the assignment uniformly. Some patterns, such as long debugging pauses, could indicate difficulty resolving errors, as seen by Dong et al. (2021). Other students exhibited short coding durations and few submissions, suggesting efficient completion or disengagement. One limitation is the varying sample size across features. Time between submissions could not be calculated for every student, since it requires at least two submissions. Additionally, time-of-day patterns, while useful, may reflect external factors such as schedules rather than factors of struggling. Despite this, our EDA demonstrates that submission-level behaviors contain measurable structure that could inform models. These extracted features appear interpretable, making them candidates.

Future work will focus on building an explainable machine-learning model using the engineered features identified in this analysis. By utilizing an explainable model, we will help instructors understand why students are flagged as struggling and provide insight into which features indicate struggle. Expanding the dataset and validating against instructor-identified cases of struggle will strengthen the computer system's reliability. Ultimately, our goal is to design an early-warning tool to increase pass rates for the computing courses.

References

Acuña, R., & Bansal, A. (2024). WIP: Characterizing Student Programming Activity. 2024 IEEE Frontiers in Education Conference (FIE), 1–5.

Dong, Y., Marwan, S., Shabrina, P., Price, T., & Barnes, T. (2021). Using Student Trace Logs To Determine Meaningful Progress and Struggle During Programming Problem Solving. Proceedings of The 14th International Conference on Educational Data Mining (EDM).

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. Á., Sheard, J., Skupas, B., Spacco, J., Szabo, C., & Toll, D. (2015). Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. Proceedings of the 2015 ITiCSE on Working Group Reports, 41–63.

Messer, M., Brown, N. C. C., Kölling, M., & Shi, M. (2024). Automated Grading and Feedback Tools for Programming Education: A Systematic Review. ACM Transactions on Computing Education, 24(1), 1–43.

Tabarsi, B. T., Limke, A., Reichert, H., Qualls, R., Price, T., Martens, C., & Barnes, T. (2022). How to Catch Novice Programmers' Struggles: Detecting Moments of Struggle in Open-Ended Block-Based Programming Projects using Trace Log Data.